REMARKS

Claims 1-22 are in this case. Claims 1-22 have been rejected. Claims 6-9, 15 and 16 have been rejected. Claims 6 and 15 have now been amended. Formal drawings are submitted, including the requested changes to Figures 1 and 8. The requested amendments to the Brief Description of the Drawings and page 13 have been made. A new Oath and Declaration is enclosed.

REJECTIONS UNDER 35 U.S.C. 112

The Examiner has rejected claims 6-9, 15 and 16 under 35 USC second paragraph as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention. Applicant respectfully traverses the rejections of the Examiner.

The Examiner has specifically rejected claim 6 (and dependent claims 7-9) for reciting "computing a trigger for each process", as "each process" is stated to be indefinite.

The Examiner also stated that the limitation "wherein the verification language features symmetry" is indefinite for lack of an antecedent basis.

While continuing to traverse the rejections of the Examiner, Applicant has chosen to amend claims 6 and 15 as follows to expedite the prosecution. Claim 6 now reads "each process of the code in the first language", while Claim 15 now reads "wherein the first language" in place of "wherein the verification language".

Applicant feels that these amendments overcome the rejections of the Examiner in this regard.

REJECTIONS UNDER 35 U.S.C. 102(e) – Edwards

The Examiner has rejected claims 1-22 under 35 U.S.C. 102(e) over US Patent No. 6,625,797 to Edwards et al. ("Edwards"). The rejections of the Examiner are respectfully traversed.

Edwards teaches a method for efficient compiling software for various hardware implementations. This method is stated to allow the software programmer to work at a high level of abstraction, while the machine-dependent (hardware-dependent) details of the program are handled according to the taught method, without requiring direct intervention by the programmer. The programmer constructs the program according to a high level of abstraction, while a semantic model handles the hardware-dependent aspects of the program. The semantics are interpreted through a tool which creates a control and flow data graph as part of the compilation of the software. The method of Edwards is specifically taught as being useful for such computer programming languages as C and C++, which are high level software languages.

By contrast, the present invention is of a system and method for synthesizing a verification language for compilation into a target language. This method enables the underlying control structure of the verification language to be determined, and then used to map the dynamic behavior of the verification language onto the target

language as part of a static framework. Verification languages have a number of characteristics which make such a process difficult. For example, these languages feature structural constraints, dynamic behavior, and a hierarchy of objects, as stated in claim 1. Structural constraints determine the relationships between the objects in the hierarchy of objects, and therefore need to be properly handled in order to map the dynamic behavior of the verification language for conversion to the target language.

This process is not simple, because the present invention is able to map the dynamic behavior as part of a static framework, also as stated in claim 1, without losing any of the functionality or capabilities of the starting (verification) language. The presence of multiple objects and multiple structural constraints governing the relationships between the objects increases the difficulty of this process, because of the need to capture temporal, dynamic behavior in the static framework, without losing any of the features of the verification language.

The first stage in preferred embodiments of the present invention is the elaboration of an instance tree. This is a key process as described with regard to Figure 1 and the accompanying textual description. This elaboration is done based on structural constraints. The elaboration of the instance tree enables the method of the present invention to handle a plurality of objects, each having some associated storage and some function. This aspect is important, because of the behavior of the objects and the constraints; not every constraint may be resolvable at every level, as described with regard to Figure 1. The method also optionally includes construction of an elaboration graph (during this process) and a data/control graph. The construction of

both graphs is preferred because of the nature of the verification language as containing a hierarchy of objects, with structural constraints determining relationships between the objects.

There are therefore a number of clear and crucial differences between Edwards and the present invention. For example, Edwards does not teach a method for compiling languages which feature "structural constraints". The languages described in Edwards do not have such constraints. Edwards gives examples such as C++ and Java, which are high-level software languages. High-level software languages do not feature constraints and/or do not feature multiple objects with relationships determined by constraints. These software languages are not equivalent to verification languages, which have a number of defined characteristics: structural constraints, dynamic behavior, and a hierarchy of objects. In order for the method of the present invention to be operative, it must be able to handle languages with these characteristics, as stated in claim 1.

By contrast, the method of Edwards can only handle programs with a single object, rather than multiple objects, because Edwards fails to show how to determine which object instances exist in the system for a program being compiled. This fundamental omission limits the applicability of the taught method of Edwards to a single designated object. Therefore, the method of Edwards would not be operative for languages as defined in claim 1, because this method cannot handle multiple objects. The method of Edwards does not and cannot handle such issues as determining which instances exist, interaction patterns between independent objects,

race detection, scheduling etc, because of the inability of Edwards to handle multiple objects.

By contrast, the method of the present invention is able to handle the complexity arising from a multiplicity of objects, for example by determining the elaborated roles (storage), deriving the access patterns (creation of the conflict graphs), conflict resolution and scheduling. The method of Edwards cannot handle these issues and would therefore be inoperative for the languages as defined in claim 1 of the present application.
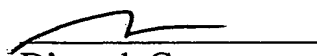
The present invention also optionally features an elaboration graph as well as a data/control flow graph, while Edwards only discloses the latter and indeed would only be operative with the latter type of graph.

In addition, the static framework of resources in claim 1 is not equivalent to a hardware-implementation independent flowgraph as taught by Edwards, again with regard to the inability of Edwards to handle multiple objects. Furthermore, substituting nodes in the graph of Edwards is not equivalent or even similar to mapping dynamic behavior of the second language in claim 1.

Therefore, the method of the present invention, as recited in the claims of the present application, is clearly different from the method of Edwards, which would be inoperative with languages as defined in claim 1. Thus, claims 1-22 are clearly novel and non-obvious over Edwards.

From the above remarks and amendments, Applicant feels that claims 1-22 are now in condition for allowance. Prompt notice of allowance is respectfully and earnestly solicited.

Respectfully submitted,

D'vorah Graeser
US Patent Agent
Reg. No. 40,000

Date: April 21, 2004